AMENDMENTS TO THE SPECIFICATION

In the Specification:

Please replace paragraphs [0001], [0042], [0045], [0109], [0152], and [0163] of the specification with the following replacement paragraphs [0001], [0042], [0045], [0109], [0152], and [0163]:

[0001] This application is related to the following co-pending U.S. Patent Applications: ~~Ser. No.~~ U.S. Publication No.[[_____]] 7,103,874 ~~(Atty. Dkt. No. MSFTP518US)~~ entitled "MODEL-BASED MANAGEMENT OF COMPUTER SYSTEMS AND DISTRIBUTED APPLICATIONS" filed on October [[___]] 23, 2003; ~~Ser. No.~~ U.S. Publication No. [[_____]] 20050114494 ~~(Atty. Dkt. No. MSFTP519US)~~ entitled "SCALABLE SYNCHRONOUS AND ASYNCHRONOUS PROCESSING OF MONITORING RULES" filed on October [[___]] 24, 2003; ~~Ser. No.~~ U.S. Publication No. [[_____]] 20050091640 ~~(Atty. Dkt. No. MSFTP520US)~~ entitled "RULES DEFINITION LANGUAGE" filed on October [[___]] 24, 2003; and, ~~Ser. No.~~ U.S. Publication No. [[_____]] 20050091635 ~~(Atty. Dkt. No. MSFTP522US)~~ entitled "USE OF ATTRIBUTION TO DESCRIBE MANAGEMENT INFORMATION" filed on October [[___]] 23, 2003.

[0042] Referring now to FIG. 3, there is illustrated a block diagram of a system where an instrumentation catalog is generated using attribution and the URI architecture of the present invention. There is provided a client 300 and a web host 302 demarcated by a machine boundary 304. The client 300 includes a software client API DLL (Dynamic Link Library) 306, which is an executable program module that facilitates communication 308 (*e.g.*, web services, denoted as WS) with the web host 302. The web host 302 includes a host client API DLL 310 that facilitates communication with a number of resident processes 314 (also called providers). The host client API 310 can interface to the respective providers 314 in a number of ways, which provider processes 314 are demarcated from the web host 302 by a demarcation line 316. The separate host providers 314 include a programming language application provider 318 (*e.g.*, C#), a native service provider 320, and a classic[[a]] provider 322. Note, however, that the application provider 318 can be any suitable programming application.

[0045] There are several different provider scenarios that can be addressed, which include the application provider 318 that was just described, the native service provider 320, and the classic provider 322. The native service 320 is written in unmanaged code (or native code). Thus, managed code needs to be "wrapped" around the native code such that the native service 320 can be suitable for management in accordance with the present invention. The native service 320 communicates with the host DLL 310 using WIN32 calls 334 over an IPC (InterProcess Communication) link ~~336~~ 337. In this case, the health information of the native service 320 is instrumented by attributing the managed code wrapper 336 facilitating communication of the health information to the client 306.

[0109] The following sample URI returns a collection of all running processes and gets the process object that represents the OS process with process id=12.

[0152] Referring now to FIG. 4, there is illustrated a model-based management architecture 400 that utilizes the URI architecture of the present invention. The model-based management approach allows a developer to describe an application or service 402 in terms of its constituent components and desired states in terms of functionality, configuration, security, and performance. Thus, an application or service description 404 facilitates describing the application 402 in terms of one or more components, including at least a tasks component ~~406~~ 412, a system component ~~408~~ 410, a manifest component ~~410~~ 408, a model component ~~412~~ 406, and an attribution component 414. A computer system 416 uses the application description 404 at installation of the application 402 to configure management services 418 associated with the computer operating system. The management services then help ensure uptime of the application or service 402 through automatic management actions such as configuration management, problem detection, diagnosis, and recovery. The model also describes common tasks that the administrator may perform. The model-based management architecture 400 facilitates a lower total cost of ownership, and is used across the application lifecycle from development, to deployment, operations, and business analysis.

[0163] The desired states of the administrator can be expressed in the code, which is surfaced in the manifest and passed for monitoring by the monitoring service. The system can, based upon instructions in the manifest, monitor the application or service and alert the administrator when the application or service does not meet the performance, and based on the instructions, take corrective actions. For example, where a test setting for e-mail is not maintained, but falls below a threshold for a period of time, another machine can be added until the load subsides, and the network traffic can also be used as a trigger increasing the number of resources to handle the given load. A goal is to automate as much [[a]] as possible so that the administrator is involved only when manual action is required.